WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Friday, February 06, 2004

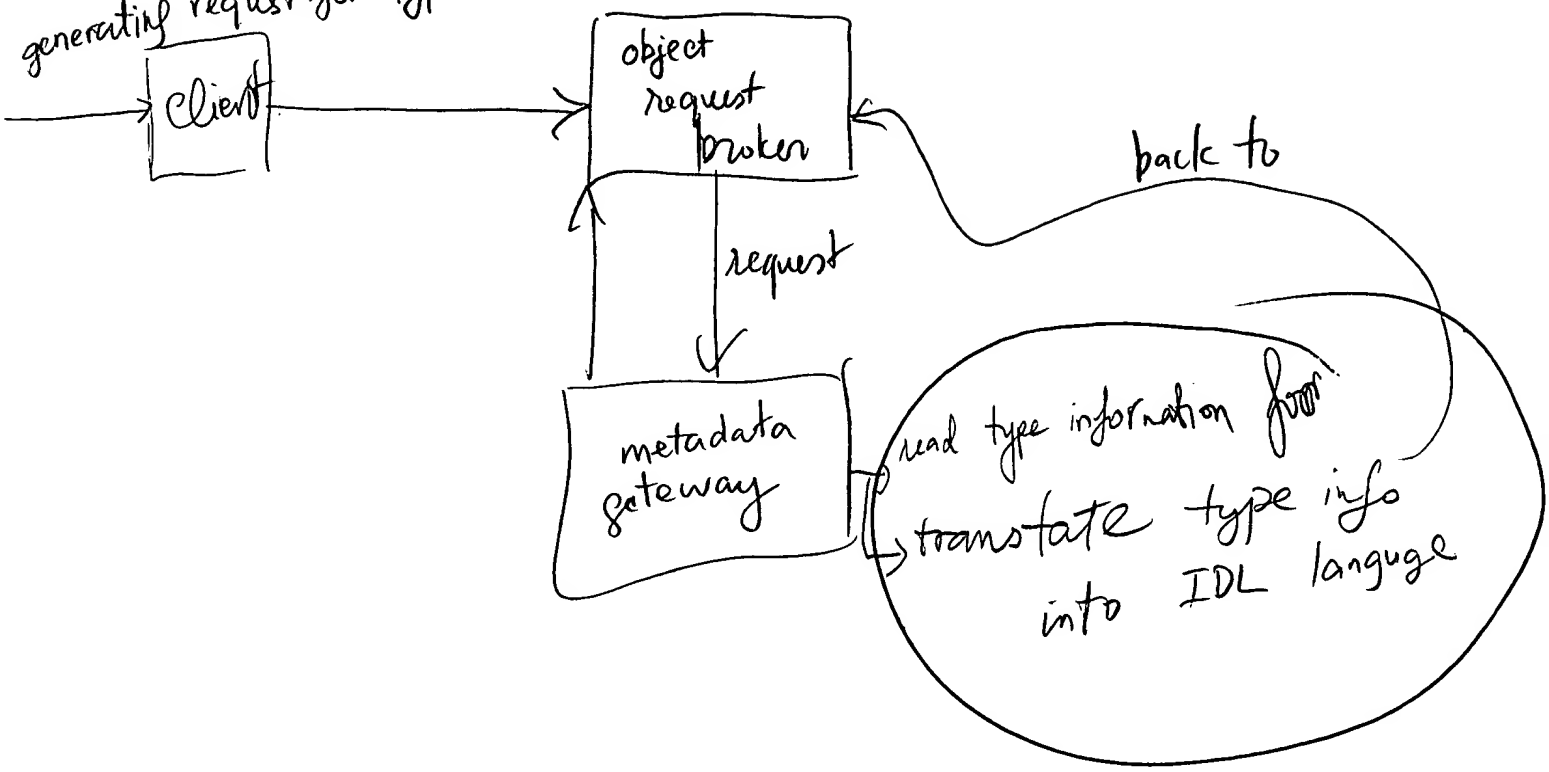
Hide?	Set Name	Query	Hit Count
		DB=USPT; PLUR=YES; OP=ADJ	
☐	L12	L11 and format	24
☐	L11	L9 and (database or data base)	29
☐	L10	L9 and cobra	2
☐	L9	l6 and platform	41
☐	L8	l1 and network and L6	1
☐	L7	l1 and netqoek and L6	0
☐	L6	l4 same L5	61
☐	L5	idl or interface definition language	1429
☐	L4	(translat$4 or convert$4) same (format or type)	197690
☐	L3	client and server	19977
☐	L2	translat$4 and L1	1
☐	L1	6496833.pn.	1

END OF SEARCH HISTORY

h e b b cg b chh e b f c e h

Manging a network

generating request for type info



First Hit Fwd Refs

End of Result Set

Generate Collection Print

L2: Entry 1 of 1 File: USPT Dec 17, 2002

DOCUMENT-IDENTIFIER: US 6496833 B1
TITLE: System and method for generating code for query object interfacing

Brief Summary Text (9):
Several prior art methods of implementing the three-tiered model exist, however, most existing DBMS access mechanisms and tools, including fourth generation languages (4GLs) and application programming interfaces (APIs), have been designed for the two-tiered model and are ill-suited for use in the three-tiered model. Consequently, several prior art designs including "database" objects and "active data" objects have their own strengths and drawbacks. One promising prior art approach to constructing a middle tier containing business logic uses "query objects." Each query object is a server object that: (1) translates client method invocations into equivalent queries in a query language which is understood by a database; (2) issues those queries to the database; and (3) returns the results as strongly-typed data values.

Brief Summary Text (11):
Each query object provides as part of its interface one or more parameterized methods and calls on each method are translated by the query object into one or more standard queries such as SELECT, UPDATE, INSERT and DELETE or into the initiation of one or more stored procedures in the database. In order to use the query object, a client first establishes a connection to the query object via some mechanism, such as the CORBA naming service. One of the query object's methods is then invoked and the query object then executes the query.

Brief Summary Text (12):
However, in order to operate properly, the query object must be constructed to generate the correct DBMS queries in response to client requests. Constructing a query object to generate the correct queries requires a knowledge of SQL, an understanding of the underlying database schema, the possible handling of intermediate results generated by the query and interpretation of the results. In addition, a query object developer must consider other issues such as connection to the database using CORBA or similar arrangement, concurrency problems and translation required between the interface used by the query object and the API used by the database. Consequently, many query objects, including the most general query objects, are hand-written by skilled and knowledgeable developers. Hand-written objects are difficult to maintain and may require rewriting if the database schema changes. Therefore, it would be desirable to automate the generation of query objects.

Detailed Description Text (6):
More particularly, query objects 208, 214 are server objects that translate client requests into appropriate DBMS queries, issue those queries to the DBMS systems and return the results to the client. Functionally, each query object 208, 214 creates a logical "wrapper" that encapsulates a specific, application-dependent set of queries and provides the results to its clients as strongly typed values. Each query object 208, 214 is associated with a single DBMS system 218, 222, respectively, and has the capability of interacting with specific DBMS query APIs

(represented by arrows 216 and 220) and parameters for the queries. However, the query object provides a DBMS-independent API (represented by arrows 210 and 212, respectively) to the business object 206 for accessing the databases 224, 226. Each query object also provides a mechanism for managing connections to the DBMS systems 218, 222 associated with the databases 224, 226, including methods for locating the databases 224, 226 in the distributed environment, methods for logging on to the DBMS systems 218, 222 and facilities for managing the connection resources. A query object may also provide multi-threading support, if necessary.

Detailed Description Text (8):
The query objects 208, 214 encapsulate expertise about the underlying databases, including the DBMS specific query languages used, particular DBMS APIs, database schemas, handling of intermediate query results and a possibly non-trivial interpretation of results. Functionally, query objects 208, 214 translate domain-specific questions and operations into the appropriate DBMS-specific queries that manipulate the databases 224, 266 and provide results. The full power of each DBMS engine (not shown) is available, including its sorting and indexing techniques and query language processors and optimizers since the queries work directly on the databases 224, 226 themselves. Moreover, the query objects 208, 214 can utilize knowledge about the particular DBMS engine being used and any special optimization tricks.

Detailed Description Text (28):
FIG. 7 illustrates the classes which are part of the query object internal state 602 of query object generator tool 600. These classes include a QueryObjectInternalState class 700 from which an object can be instantiated, which object communicates between the user GUI and the generator code. A QueryInfo class 702 is also included which contains methods that translate parameter notations between a notation entered by the user and the notation actually used by the underlying database. For example, parameters in a query written in accordance with SQL language usually use a colon (:) to designate a parameter whereas a JDBC database driver expects a question mark (?) to designate a parameter. The QueryInfo class 702 contains methods that translate between these two notations.

Detailed Description Text (46):
In either case, the routine ends up at step 912 where the query commands entered by the user are translated into DBMS specific query language commands. This is generally accomplished by using lookup tables to translate specific commands from the input SQL language to the output DBMS specific language.

Detailed Description Text (47):
Next, in step 914, the translated query syntax is verified. This can be accomplished in a number of ways. For example, the aforementioned database schema access query object can be used to verify the validity of the translated query syntax by applying the query to the underlying DBMS engine. In some cases, the underlying DBMS API may provide a "verify" function which can be invoked to verify if the syntax of the query is correct. In other cases, the execution of a "prepare query" statement may cause the DBMS engine to verify the query. In other cases, the query may actually have to be submitted to the database. In the case of UPDATE or DELETE queries, a rollback operation may then have to be performed to "undo" the operation of the query, should it succeed.

Detailed Description Text (48):
In any case, the routine proceeds to step 916 where a determination is made whether the translated query syntax is proper or not. If the syntax is not proper, the routine returns to step 906, possibly generating an error message, to indicate the user that the query is improper and must be rewritten.

Detailed Description Text (50):
The steps involved in building a parameter list are illustrated in FIG. 10. This

routine starts in step 1000 and then proceeds to step 1002 where the SQL query text string is first parsed into tokens using a conventional lexical analyzer which examines the query string and breaks it up into tokens utilizing predetermined delimiters such as spaces. The tokens are then examined for a parameter identifier which may, for example, consist of colons (:) or at (@) signs that identify parameters. In step 1004 each parameter is translated into the correct underlying DBMS syntax.

CLAIMS:

1. A system for generating interface definitions and source code which implement a query object for accessing a database using a predetermined programming language in a computer having a memory, the system comprising: an input mechanism which receives an input query from a user; a query translator which translates query commands in the input query to database commands used with the database; and at least one code generator class in the memory, which contains methods which generate the source code required to implement the query object, the code generator class generating source code that is specific to the database to be accessed and the programming language and that uses the database commands to generate the interface definitions and the source code.

12. A method for generating interface definitions and source code which implement a query object for accessing a database using a predetermined programming language in a computer having a memory, the method comprising the steps of: (a) receiving an input query from a user; (b) translating query commands in the input query to database commands used with the database; and (c) creating a plurality of code generator classes in the memory, each of which contains methods which generate the source code required to implement the query object, the code generator classes generating source code that is specific to the database to be accessed and the programming language and that uses the database commands to generate the interface definitions and the source code.

23. A computer program product for generating interface definitions and source code which implement a query object for accessing a database using a predetermined programming language in a computer having a memory, the computer program product comprising a computer usable medium having computer readable program code thereon including: program code which receives an input query from a user; program code which translates query commands in the input query to database commands used with the database; and a plurality of code generator classes, each of which contains methods which generate the code required to implement the query object, the code generator classes generating source code that is specific to the database to be accessed and the programming language and uses the database commands to generate the interface definitions and the source code.